An efficient iterative source routing algorithm

Gang Cheng, Ye Tian, and Nirwan Ansari
Advanced Networking Lab, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102
{gc2, yt2, Ansari}@njit.edu

Abstract: Finding a feasible path subject to multiple constraints in a network is an NP-complete problem and cannot be solved in polynomial time. Thus, many proposed source routing algorithms tackle this problem by transforming it into the shortest path problem, which is P-complete, with an integrated cost function that maps multi-constraints into a single cost. However, this approach may fail to find a feasible path even when it exists. Some algorithms improve their success ratio of finding a feasible path by performing multiple shortest path searches; each search is associated with a different cost function. Thus, how to vary the cost function is a critical issue for this kind of approach. In this paper, we propose an efficient algorithm to expedite the multiple shortest path searches based on the analysis of the cost function. Simulations show that our algorithm outperforms its contender in terms of computational complexity, and success ratio of finding a feasible path.

1. Introduction

One of the challenging issues for high-speed packet switching networks to facilitate various applications is to select feasible paths that satisfy different quality-of-service requirements. This problem is known as QoS routing. In general, two issues are related to QoS routing: state distribution and routing strategy [1]. State distribution addresses the issue of exchanging the state information throughout the network [2]. Routing strategy is used to find a feasible path that meets the QoS requirements. In this paper, we focus on the latter task, and assume that accurate network state information is available to each node. A number of research works have also addressed inaccurate information [3–7], which is, however, beyond the scope of this paper.

QoS constraints can be categorized into three types: concave, additive, and multiplicative. Since concave parameters set the upper limits of all the links along a path such as bandwidth, we can simply prune all the links and nodes that do not satisfy the QoS constraints. We can also convert multiplicative parameters into additive parameters by using the logarithm function. For instance, we can take -log(1 - p) as the replacement for loss rate \( p \). Thus, we focus only on additive constraints in this paper. It has been proved that multiple additively constrained QoS routing is NP-complete [8]. Hence, tackling this problem requires heuristics. In [9], a heuristic algorithm was proposed based on a linear cost function for two additive constraints; this is a \( MCP \) (Multiple Constrained Path Selection) [1] problem with two additive constraints. A binary search strategy for finding the appropriate value of \( k \) in the linear cost function \( w_1(p) + kw_2(p) \) or \( kw_1(p) + w_2(p) \), where \( w_i(p) \) \( (i = 1, 2) \) are two respective weights of the path \( p \), was proposed, and a hierarchical Dijkstra algorithm was introduced to find the path. It was shown that the worst-case complexity of the algorithm is \( O(log B(m + n\log n)) \), where \( B \) is the upper bound of the parameter \( k \), \( m \) is the number of links, and \( n \) is the number of nodes. The authors in [10] simplified the multiple constrained QoS routing problem into the shortest path selection problem, in which the Weighted Fair Queuing (WFQ) service discipline is assumed. Hence, this routing algorithm cannot be applied to networks where other service disciplines are employed. Similar to [9], Lagrange Relaxation Based Aggregated Cost (LARAC) was proposed in [11] for the Delay Constrained Least Cost path problem (DCLC). This algorithm is based on a linear cost function \( c_i = c + \lambda d \), where \( c \) denotes the cost, \( d \) the delay, and \( \lambda \) an adjustable parameter. It differs from [9] on how \( \lambda \) is defined: \( \lambda \) is computed by Lagrange Relaxation instead of the binary search. It was shown that the computational complexity of this algorithm was \( O(m^2 \log m) \).

However, in [12], for the same problem (DCLC), a non-linear cost function was proposed after considering the shortcoming of the linear cost function. Many researchers have posed the QoS routing problem as the k-shortest path problem, but the computational complexity is generally very high [13,14]. To solve the delay-cost-constrained routing problem, Chen and Nahrstedt proposed an algorithm [15], which maps each constraint from a positive real number to a positive integer. By doing so, the mapping offers a “coarser resolution” of the original problem, and the positive integer is used as an index in the algorithm. The computational complexity is
reduced to pseudo-polynomial time, and the performance of the algorithm can be improved by adjusting a parameter, but with a larger overhead.

As reviewed above, some proposed algorithms perform multiple shortest path searches by varying cost function. Note that the execution of any shortest path searching algorithm, such as Dijkstra and Bellman-Ford algorithm, gets a same path if the same cost function is used and there is only one shortest path for a specific network. Thus, the way of changing the cost function directly affects the performance of the corresponding multi-constrained routing algorithm. In this paper, we propose a source routing algorithm that adjusts the cost function iteratively and efficiently. Through simulations, the performance of our algorithm outperforms its contenders in terms of success ratio and computational complexity.

2. Problem formulation and notations

Definition 1: Multiple Additive Constraints Path Selection (MACP): Assume a network is modeled as a directed graph $G(N,E)$, where $N$ is the set of all nodes and $E$ is the set of all links. Each link connected from node $u$ to $v$, denoted by $e_{uv} = (u,v) \in E$, is associated with $M$ additive parameters: $w_i(u,v) \geq 0$, $i = 1, 2, ..., M$. Given a set of constraints $(c_1, c_2, ..., c_M)$ and a pair of nodes $s$ and $t$, find a path $p$ from $s$ to $t$ subject to $W_i(p) = \sum f_{e_{uv}} w_i(u,v) \leq c_i$, $i = 1, 2, ..., M$ is a feasible path.

Definition 2: Any path selected by MACP is a feasible path, that is, any path $p$ from $s$ to $t$ that meets the requirement, $W_i(p) = \sum f_{e_{uv}} w_i(u,v) \leq c_i$, $i = 1, 2, ..., M$, is a feasible path.

Notations:
- $f(\chi)$: Cost function, where $\chi = (x_1, x_2, ..., x_M)$.
- $C$: The vector representation of the QoS constraints $(c_1, c_2, ..., c_M)$.
- $W(p)$: The weight vector of path $p$, i.e., $(W_1(p), W_2(p), ..., W_M(p))$, where $W_i(p) = \sum f_{e_{uv}} w_i(u,v)$.
- $C(p)$: The cost of path $p$, which can be written as $\sum f(w_{e_{uv}}) \sum f_{e_{uv}} w_i(u,v)$, where $f()$ is the cost function.
- Note that $C(p) \neq f(W(p))$ because $f(W(p)) = f(\sum f_{e_{uv}} w_{e_{uv}}(u,v) \sum f_{e_{uv}} w_i(u,v) \sum f_{e_{uv}} w_i(u,v))$.
- However, if $f(\chi)$ is linear, $C(p) = f(W(p))$. It should also be noted that, to be a cost function, $f(\chi)$ should have the property that $\frac{\partial f(\chi)}{\partial x_i} \geq 0$ if $x_i \geq 0$, $i = 1, 2, ..., M$; i.e., the cost function is increasing with respect to each additive parameter.

3. The proposed algorithm

Since multiple QoS constraints routing problem is NP-complete, no algorithms can ensure that the termination condition can be met in polynomial time. Here, the termination condition is referred to as any of the following conditions:
1. A feasible path is found.
2. It is certain that no feasible path exists.

Thus, like [9], multiple searches which incorporate an algorithm for adjusting the cost function parameters are necessary in order to increase the success ratio of finding a feasible path.

We use the following cost function as the initial cost function:
\[ f(\chi) = \sum_{i=1}^{M} c_i \]

Let assume the function for the $k^{th}$ search is $f^{k+1}(\chi, x_1, ..., x_M) = \sum \beta_i^{k+1} x_i$, $\beta_i^{k+1} \geq 0$, $i = 1, 2, ..., M$.

Thus, $\beta_i^{k+1} = \frac{1}{c_i}, i = 1, 2, ..., M$.

We shall next present some theorems showing the motivation behind our algorithm.

Theorem 1: No feasible path exists if the least cost path of the $k^{th}$ search has the cost no less than $f^{k+1}(\chi)$.

Proof: By contradiction. Assume path $\hat{p}$ satisfies the constraint $C$ and the least cost among all paths is no less than $f^{k+1}(\chi)$; that is, $C(p) \geq f^{k+1}(\chi), \forall p \Rightarrow C(p) \geq f^{k+1}(\chi)$

Also $\frac{\partial f^{k+1}(\chi, x_1, ..., x_M)}{\partial x_i} = 0 \Rightarrow f^{k+1}(W(\hat{p})) = C(\hat{p})$

Thus, $f^{k+1}(W(\hat{p})) \geq f^{k+1}(\chi)$

However, since $\frac{\partial f^{k+1}(\chi)}{\partial x_i} \geq 0$ and path $\hat{p}$ satisfies the constraint $C$, $W(\hat{p}) < c_i, \forall i \in [1, 2, ..., M]$.

$f^{k+1}(W(\hat{p})) > f^{k+1}(\chi)$, which contradicts $f^{k+1}(W(\hat{p})) \geq f^{k+1}(\chi)$, and thus Theorem 1 is proved.

Lemma 1: Path $p$ is a feasible path only if $f^{k+1}(W(p)) \leq f^{k+1}(\chi)$, $k = 1, 2, ..., M$.

Proof: This can be readily derived from Theorem 1.

Based on Theorem 1 and Lemma 1, a well-designed algorithm for adjusting the cost function
referred to as QA (Quick-Adjusting Algorithm) is proposed to expedite reaching the termination condition.

We first execute the shortest path searching algorithm with the cost function \( f^0(x) = \sum_{i=1}^{M} \beta_i x_i \) (\( \beta_i = c_i^{-1} \), \( i = 1, 2, \ldots, M \)) as the initial cost function. Thus, there are 3 possible outcomes:

1. A feasible path is found.
2. A feasible path is found and the least cost path \( p_1 \) satisfies \( C(p_1) \leq f^1(x) \).
3. A feasible path is not found and the least cost path \( p_1 \) satisfies \( C(p_1) < f^1(x) \).

For the first case, the termination condition is met, and the search terminates. For the second one, the least cost path \( p_1 \) satisfies \( C(p_1) \geq f^1(x) \), and thus for any path \( p \), \( C(p) \geq f^1(x) \). By Theorem 1, no feasible path exists (assuming that the probability of having another least cost path that is feasible with cost exactly equal to \( f^1(x) \) is negligibly “zero” if not zero), the search terminates here. For the third case, the shortest path searching algorithm needs to be executed again with a different cost function. With the new cost function \( f^1(x) = \sum_{i=1}^{M} \beta_i x_i \), there are 4 possible outcomes for this new search:

1. A feasible path is found.
2. A feasible path is not found and the least cost path \( p_2 \) satisfies \( C(p_2) \geq f^2(x) \).
3. A feasible path is not found and the least cost path \( p_2 \) satisfies \( C(p_2) < f^2(x) \) and \( p_1 \neq p_2 \).
4. A feasible path is not found and the least cost path \( p_2 \) satisfies \( C(p_2) < f^2(x) \) and \( p_1 = p_2 \).

Here, \( C(p) \) represents the cost of path \( p \) using cost function \( f^1(x) \). Similar to the first search, the search terminates for the first two cases. However, since \( p_1 \) is the least cost path in the first search, \( p_1 \) may be the least cost path again (case 4). Nevertheless, if there exists a feasible path, case 4 will not occur. This is because \( \beta_i \)’s are set such that \( C^i(p_1) = f^i(x) \); if \( p_2 = p_1 \), then \( C^i(p_2) = f^i(x) \), implying that no feasible paths exist according to Theorem 1.

Similarly, if after \( k < M \) searches, a feasible path is not found, \( \beta_i \)’s are set such that \( C^i(p_2) = C^i(p_3) = \ldots = C^i(p_k) = f^i(x) \). Here, \( p_1 \) is a least cost path of the \( j \)th search, \( C^i(p) \) is the cost of path \( p \) with \( f^i(x) \) as the cost function, and \( f^i(x) \) is the cost function for the \((k+1)^{th}\) search. An example of how to compute \( \beta_i \) is given later. Using this procedure, there are only 3 possible outcomes for the \((k+1)^{th}\) search:

1. A feasible path is found.
2. A feasible path is not found and the least cost path \( p_{k+1} \) satisfies \( C^i(p_{k+1}) \geq f^i(x) \).
3. A feasible path is not found and the least cost path \( p_{k+1} \) satisfies \( C^i(p_{k+1}) < f^i(x) \) and \( p_{k+1} \neq p_i, i = 1, 2, \ldots, k \).

Obviously, assuming there exists a feasible path, the larger the \( k \) is, the less possible case 3 will occur, because each iteration eliminates one path, resulting in a continuous decrease in the search space. After a few iterations, only a limited number of paths are left in the search space from the source to the destination. Thus, with this method, we can gradually increase the possibility of case 1 or 2, i.e., this method can speed up the occurrence of the termination condition.

When \( k \geq M \), the solution (\( \beta_i \)’s) that satisfies all the \( k \) linear equations \( C^i(p) = f^i(x) \), \( i = 1, 2, \ldots, k \), may not exist, i.e., it is over-determined. Thus, we only need to conduct at most \( M \) searches using the above procedure. However, it is possible that a feasible path exists but cannot be found by the above method. For example, finding a feasible path for the network shown in Fig. 1 with constraint (1,1) and \( f(x,y) = x + y \) as the initial cost function will fail using the above method. Therefore, we can loosen our restriction to circumvent this case for the last iteration, i.e., \( k = M \). That is, when \( k = M \), if a feasible path is still not found, \( \beta_i \)’s are set such that \( C^i(p_{k+1}) = C^i(p_1) = \ldots = C^i(p_k) \) instead of \( C^i(p_{k+1}) = C^i(p_1) = \ldots = C^i(p_k) = f^i(x) \). By doing so, the feasible path in the last example can be found. The pseudo-code for the QA algorithm is shown in Fig. 2.

Assume the computational complexity of the shortest path search algorithm is \( O(\alpha) \), where \( \alpha \) is usually a function of the number of nodes and links of the network specific to this search algorithm. Thus, the overall computational complexity of our

![Network topology](image-url)
Algorithm QA(G,s,t,C)
1 Initial k=0, $\beta_k^i = \frac{1}{c_i}$ and $f^k(\mathbb{Q}) = \sum_{i=1}^{n} \beta_k^i c_i$
2 while $(k \leq M)$
3 $k = k + 1$
4 Execute Shortest Path Search with cost function $f^{k-1}(\mathbb{Q})$ and get the shortest path $p_k$
5 if $p_k$ is a feasible path
6 return SUCCESS
7 else if $C^{k-1}(p_k) > f^{k-1}(\mathbb{Q})$
8 return CFAIL (by Theorem 1)
9 else if $k < M$, compute $\beta_k^i$ to make
10 $C^k(p_k) = f^k(\mathbb{Q})$ $(i, j = 1, 2, ..., k)$
11 else if $k = M$, compute $\beta_k^i$ to make
12 $C^* (p_k) = C^*(p_k)$ $(i, j = 1, 2, ..., k, i \neq j)$
13 end if
14 end while

Fig. 2. The QA algorithm.

We shall illustrate how to compute $\beta_k^i$ s in QA for the case of two additive constraints. Assume the least cost path of the first search is not a feasible path, and $(w_1, w_2)$ are its weights. Thus, in order to achieve $C^1(p_k) = f^1(\mathbb{Q})$, for the case that $w_i \neq c_i$, let $\beta_1^i = \beta_0^i$ and $\beta_2^i = \beta_1^i (w_i - c_i) (c_i - w_i)$. For the case of $w_i = c_i$, we can set $\beta_1^i = 0$ and $\beta_2^i = 1$. Thus, it can be observed that $f^1(w_1, w_2) = f^1(c_1, c_2)$.

Similar to the above, it can be proved that $\beta_k^i > 0$. So, for the case of two additive constraints, the computational complexity introduced by computing $\beta_k^i$ s is trivial and negligible as compared to the overall computational complexity. For the general case, i.e., multiple additive constraints, any set of $\beta_k^i$ s can be chosen, as long as $C^k(p_k) = f^k(\mathbb{Q})$.

4. Simulations

We evaluate our algorithm by incorporating QA with the Dijkstra algorithm and comparing it with the algorithm in [9], which is in the same category of our algorithm and the best so far reported in the literature. The network topology (Fig. 3) presented in [9] and [15] is adopted for comparison purposes. In the simulations, the link weights are independent and uniformly distributed from 0 to 1, and all data are obtained by running 1,000,000 requests. The performance of all algorithms are evaluated by the success ratio (SR) defined below:

$$SR = \frac{\text{Total number of success request of the algorithm}}{\text{Total number of success request of the optimal algorithm}}$$

Fig. 3. Network Topology.

The algorithm that can always locate a feasible path as long as it exists is referred to as the optimal algorithm. Here, it is achieved simply by flooding.

In the simulations, two QoS constraints are set to be equal, and increase from 0.5 to 3.9 with an increment of 0.2.

As shown in Fig. 4, the lower bound of the success ratio of our algorithm is 99.35%, while that of Korkmaz et al.’s algorithm is 99.15%. Note that the worst case computational complexity of our algorithm is only the computational complexity of the Dijkstra Algorithm while that of Korkmaz et al.’s algorithm is $O((M + n)\log n)$, here $B = 1000$. So, our algorithm outperform Korkmaz et al.’s algorithm in terms of computational complexity and success ratio of finding a feasible path.
5. Conclusions

In this paper, we have presented an iterative source routing algorithm for solving multi-constrained QoS routing problem – QA. QA expedites the search of a feasible path in the way that it iteratively reduces the search space by varying the cost function. The key issue is that QA ensures that the previous search result, which is not a feasible path, will not be the output of the future searches if a feasible path exists. Through simulations, we demonstrate that our algorithm outperforms its contenders in terms of computational complexity and success ratio of finding a feasible path.

6. References


